AFRL-RI-RS-TR-2012-149

INVESTIGATING ARCHITECTURAL ISSUES IN NEUROMORPHIC COMPUTING

MAY 2012

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2012-149 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
ROBINSON PINO
Work Unit Manager

/s/
PAUL ANTONIK, Technical Advisor
Computing & Communications Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
MAY 2012	Final Technical Report	SEP 2010 – OCT 2011

4. TITLE AND SUBTITLE

INVESTIGATING ARCHITECTURAL ISSUES IN NEUROMORPHIC COMPUTING

5a. CONTRACT NUMBER
In House

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)

Robinson E. Pino

5d. PROJECT NUMBER
23T1

5e. TASK NUMBER
PR

5f. WORK UNIT NUMBER
OJ

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RITB
525 Brooks Road
Rome NY 13441-4505

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/Information Directorate
Rome Research Site/RITB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2012-149

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2012-2781
Date Cleared: 11 MAY 2012

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Our objective is the study of an Intelligent Text Recognition System (ITRS) that mimics the human information processing procedure to fill in the missing or damaged text by considering the word level and sentence level context. The system is built upon two cognitive computing models, the Brain-State-in-a-Box (BSB) Attractor Model and the Cogent Confabulation Model. The former performs character detection and later performs word and sentence completion. Given a scanned text image where each character is 15-by-15 pixels large, experimental results show that, when 20% of the character images are damaged by a 1-pixel-wide horizontal scratch running through the center of the image where most of the information to distinguish amongst various characters is found, the ITRS recognizes complete sentences at 92% accuracy. When 60% of the character images are damaged by a 3-pixel-wide horizontal scratch located at the center, the ITRS recognizes sentences at 64% accuracy.

15. SUBJECT TERMS
high performance computing, neuromorphic computing, confabulation, cortex emulation, PS3 array, brain state in a box

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			ROBINSON E. PINO
U	U	U	UU	29	19b. TELEPHONE NUMBER (*Include area code*) N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 SUMMARY

This effort has explored the issues associated with the efficient mapping of neuromorphic computing strategies onto advanced computational architectures. The computing performed by neurological systems produces cognitive phenomena that have been high value, yet elusive, goals of computational researchers. Neuromorphic computing, as evident in primate brains, uses massive collections of modest speed synapses and neurons operating asynchronously in parallel.

This computation is characteristically:

• Performed with precision and robustness;

• Accomplished with very low power consumption;

• Performed in real time, allowing the fusion of sensing, planning, and interaction with the environment

• Performed without programming, based on experience

A characteristic challenge of the effort is its multidisciplinary nature. It combined ideas and research from diverse fields including computer architecture, neuroscience, cognitive psychology, cognitive modeling, dynamical systems, belief systems, software and computer engineering.

This effort has produced some infrastructure suitable for continuing cortical modeling research. It consists of software, in addition to the models discussed, developed for and applied to modeling a visual input stream (a retina model, an optic chiasm model, and a thalamic-LGN model), a high throughput Publish/Subscribe messaging system, and high performance machine clusters (AFRL/RI's cluster of 1760 PS3 CELL-BE platforms with 84 nodes with dual sextet cores, 1,008 cores total).

2.0 BACKGROUND

This is a report on the work sponsored by the Air Force Office of Scientific Research and conducted at the Information Directorate of the Air Force Research Laboratory to investigate architectural issues surrounding neurobiological inspired computational methods based on networks of structures roughly emulating cortical columns. The work consisted of a three year multidisciplinary effort focusing on determining how neurological systems perform those aspects of cognition associated with sensing and perception aspects of cognition. The work has focused on ventral tract (object recognition) aspects of the brain. Dorsal tracts are parallel to ventral tracts, and are theoretically associated with spatial properties.

2.1 Rationale

The focus of interest is the development of computer architectures capable of advanced applications requiring large scale parallel computing and complex communication between nodes. The interest is driven by the value of applications which can make use of such architectures (perception and situational awareness are examples), and technology advances moving to surpass one thousand cores per die in the next few years. This technology trend is grounded in the problems of cooling, clock slew and parasitic inductances, which grow significantly as clock speeds increase. The previous rapid rate of clock speed increase for CPUs has disappeared. Chip developers have turned to multicore technology to make use of the

continuing exponential trend towards increased transistor density. Multicore technology shifts problems from hardware to software and multiplies available parallelism. To make productive use of 100 thousand to 1 million processors, one must provide software, which can efficiently harness the parallelism inherent in the hardware. Software development is labor intense. The cost intensity grows significantly as parallelism increases. Software developers have few methods available to them to deal with parallel system design, except for messaging systems and multithread programming. No significantly better methods have emerged into common practice which displace or build on these. These techniques are suitable for small scale parallelism but grow unwieldy for systems even a few thousand processors. Existing High Performance Computer (HPC) platforms, like Blue Gene/L, can be configured with more than 130K processor cores. The challenge of harnessing parallelism on that scale for all but "embarrassingly parallel" applications (an application where very little communication is needed between processes) challenges the limits of programmability. Yet neural processing effectively harnesses parallelism on at least this scale.

Cognition presents as an excellent target of study because primate brains are examples of the kind of computing architecture we seek. It also holds promise to meet the "programmability challenge" of large scale parallelism with self supervised learning, and is therefore itself potentially a key technology for approaching other difficult to scale applications like Parallel Discrete Event Simulation (PDES). PDES applications are models of physical processes in terms of state changes at discrete points in time. Example PDES applications include networking, electronics, command and control, particle physics, machinery, weather, and communication systems. These applications are characteristically intense in terms of CPU but challenge computer architectures with the need to communicate events to all affected elements within the simulation. PDES applications typically do not scale well across even a few hundred nodes. In parallel AFRL (6.2) research, new architectures are being developed to better address PDES, largely motivated by the neuromorphic insights uncovered in the research project. Specifically, the custom connectivity of synaptic networks is being mimicked in field programmable gate arrays to reduce the latency of event propagation across the massive architecture.

Beyond the rather pragmatic utility of PDES acceleration on neuromorphically inspired architectures, there are many aspects of cognition valuable to the Air Force missions which may be within near-term grasp. Some of these include learning, vision, audition and olfaction, ability to navigate an environment, and goal seeking.

These abilities have long been among the objectives of artificial intelligence research; but progress has been limited. In particular, solutions have lacked the robustness observed in natural systems. Thus competing "connectionist" approaches arise which draw inspiration from neurobiology to seek out these abilities. However, a significant impediment is that science has not yet worked out how the synapses, neurons and glia cells of a brain work systematically together to achieve cognition. Neuroscience has not traditionally been a "system centric" science. It tends to focus on ever narrowing details on anatomy, neurochemistry, and electrophysiology. While this narrowing of focus has progressed, related fields of interest have overlapped with neuroscience and provided resources crucial for the pursuit of neuromorphic computing. The medical community is an example; it relies on imaging technology for diagnosis. Neural imaging techniques are vital to neural surgeons and neurologists. Medical market pressures continue to promote improvement in resolution, and content. For example, the market has produced functional magnetic resonance imaging (fMRI) technology that can resolve blood oxygen level

dependent (BOLD) response on a sub-millimeter level, providing a means for imaging the routing of cortical nerve bundles previously undetectable in living specimens. These kinds of advances, combined with emerging large scale computer technology, enable a new approach to investigating how brains work. The new capability is the emulation of large pieces of a brain.

It is becoming feasible to emulate full scale brains on a neuron level, at least insofar as computational complexity matters. The human brain has an estimated 10^{11} neurons, each with an average estimated 104 connections to other neurons. Single neuron models need to account for synapses (connections) and somas (cell bodies). A simple synapse model uses two numerical operations (OPs): an index (address addition) and a value addition (this would be the complexity floor). A simple soma model (threshold compare and assignment) is equivalent to two OPs. Thus, a human brain emulation (if all neurons and synapses happen to fire at once; an unlikely event) would require ~$3X10^{15}$ OPs. A single Cell-BE node can peak at $2X10^{11}$ FLOPS. 15K such devices, by this measure, would be able to emulate a full sized human brain at about 1/1000 real-time speed. Certainly, synapse and neuron level models can be more complex than this estimate, but it is also true that emulation may not always need to be carried out at a low level.

Modern image processing software performs image detection and pattern recognition with fairly high accuracy given the condition that the input image is clean and fully observable. Pattern recognition becomes extremely difficult, if not impossible, when the image is partially obscured or even partially missing. Compared to computer-based image processing algorithms, the human brain exhibits extraordinary ability for pattern recognition in noisy environments because it generates anticipations based on the context of the input and knowledge of the problem.

2.2 Organization of this Report

This report is organized as follows:

- Section 3 describes the research objectives.
- Section 4 illustrates the results obtained in this effort summarizing emulation results.
- Section 5 lists references used in the document.

3.0 RESEARCH OBJECTIVES

Our objective is the study of an Intelligent Text Recognition System (ITRS) that mimics the human information processing procedure to fill in the missing or damaged text by considering the word level and sentence level context. The system is built upon two cognitive computing models, the Brain-State-in-a-Box (BSB) Attractor Model and the Cogent Confabulation Model. The former performs character detection and later performs word and sentence completion. Given a scanned text image where each character is 15-by-15 pixels large, experimental results show that, when 20% of the character images are damaged by a 1-pixel-wide horizontal scratch running through the center of the image where most of the information to distinguish amongst various characters is found, the ITRS recognizes complete sentences at 92% accuracy. When 60% of the character images are damaged by a 3-pixel-wide horizontal scratch located at the center, the ITRS recognizes sentences at 64% accuracy. Furthermore, when 10% of the characters are completely occluded, the ITRS recognizes sentences and words at more than 60% and 95% accuracy respectively. When the occluded characters increase to 30%, the sentence accuracy drops to 20% and the word accuracy drops to 85%.

4.0 TASK PERFORMANCE

This section will describe the various tasks and results performed during the effort.

4.1 Hybrid Intelligent System for Text Recognition on a Heterogeneous High Performance Computing Cluster

With the progress in high performance computing (HPC) technology, the research in machine intelligence has entered a new era. How to harness the huge amount of computing power and memory storage provided by the modern HPC clusters and convert it to useful computations that assists or even replaces the human cognition process? Will the performance of current cognitive computing model scale as the hardware resource increases? What is the bottleneck of current HPC architecture when being applied to cognitive computing and how can this be addressed by future computing tools? These are urgent questions remaining to be answered.

In previous summers, we have developed a hybrid intelligent system for text recognition. During this summer, we implemented the system on the 500 TFLOPs (Tera FLoating Point Operations Per Second) PlayStation3 cluster in AFRL/RIT. This report will introduce the basic concept of the intelligent text recognition system and the architecture of its implementation. The impact of the available hardware resources on the overall system performance will also be presented.

4.1.1 Introduction

Research discoveries in human psychology suggest that human information processing is a multi-level process [1]. Information is first processed by the sensory cortex where the complex data is reduced to abstract representations. The abstract representation is compared to stored pattern in massively parallel procedural at basal ganglia and neocortex to generate quick reaction. If more sophisticated processing such as reasoning is needed then relatively slow sequential process will occur in the prefrontal cortex. If we try to replicate these information processing steps, then it naturally requires machines with massively parallel processing capability and high computation speed as the first layer and machines with large memory space and high memory access speed at the second layer. No special requirements are needed for machines on the top layer because it processes the least amount of information among the three

layers. Furthermore, compared to our capability in pattern matching and object classification, human performs the worst in logic inference.

What has been described is very similar to the 1,800 nodes HPC cluster that has been assembled at AFRL/RITB. The cluster consists of more than 80 sub-clusters and each sub-cluster is composed of one Intel Xeon Hexa-core processor as the head node, 22 Sony PlayStation3 (PS3) computers based on IBM Cell Broadband Engine processor, and 2 NVIDIA GP-GPU. Each cell processor has one PowerPC processor and 6 synergistic processing elements (SPE). Each SPE processor is a self contained vector processor that runs 4 floating point operations at 3.2 GHz. With 6 of these SPEs, a cell processor provides 192GFlops performance. Each PS3 only has 256MB memory, which is relatively small comparing to the conventional desktop PCs. Overall, the 1,760 cell processors deliver 338 TFLOPs (Tera FLoating Point Operations Per Second) computing power and form the first layer hardware of a cognitive computing system. The Intel Xeon processor based head nodes naturally form the second layer. Each head node has 12 cores and 24GB SMART memory. The memory access speed could reach as high as 2GB/s per core.

To explore the potential of the HPC cluster, a proof of concept prototype of an intelligent system for context aware Intelligence Text Recognition is developed. It adopts neuromorphic system architecture that incorporate massive parallel high performance computing techniques with advances in artificial intelligence and human psychology. The Intelligent Text Recognition System (ITRS) reads and understands the scanned text image at high speed. It learns from what has been read and, based on the obtained knowledge; it forms anticipations and predicts the next input image (or the missing part of the current image). Such anticipation helps the system to fight with all kinds of noises that may occur.

The ITRS system is built on two cognitive computing models, the brain-state-in-a-box (BSB) model and cogent confabulation model. The details of these two models and their interactions will be presented in the next section.

4.1.2 Background

4.1.2.1 **Brain-State-In-a-Box Model (BSB)**

BSB model is a simple, auto-associative, nonlinear, energy minimizing neural network [2][3]. A common application of the BSB model is to recognize a pattern from a given noisy version. BSB model can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries [4].

There are two main operations in a BSB model, Training and Recall. In this paper, we will focus on the BSB recall operation. The mathematical model of a BSB recall operation can be represented in the following form:

$$X(t+1) = S(\alpha \cdot \boldsymbol{A} \cdot \boldsymbol{x}(t) + \lambda \cdot \boldsymbol{x}(t) + \gamma \cdot \boldsymbol{x}(0)), \qquad (4.1)$$

where:

- x is an N dimensional real vector
- ***A*** is an NxN connection matrix
- ***A·x(t)*** is a matrix-vector multiplication operation
- α is a scalar constant feedback factor
- λ is an inhibition decay constant
- γ is a nonzero constant if there is a need to maintain the input stimulation

- S( ) is the "squash" function defined as follows:

$$S(y) = \begin{cases} 1 & if\ y \geq 1 \\ y & if\ -1 < y < 1 \\ -1 & if\ y \leq -1 \end{cases} \quad (4.2)$$

In [5], we implemented and optimized the recall operation of the BSB model on the Cell Broadband Engine processor. The runtime measure shows that, we have been able to achieve about 70% of the theoretical peak performance of the processor.

4.1.2.2 **Cogent Confabulation model**

Cogent confabulation [6] is an emerging computation model that mimics the Hebbian learning, the information storage and inter-relation of symbolic concepts, and the recall operations of the brain. Based on the theory, the cognitive information process consists of two steps: learning and recall. During the learning, the knowledge links are established and strengthened as symbols are co-activated. During recall, a neuron receives excitations from other activated neurons. A "winner-take-all" strategy takes place within each lexicon. Only the neurons (in a lexicon) that represent the winning symbol will be activated and the winner neurons will activate other neurons through knowledge links. At the same time, those neurons that did not win in this procedure will be suppressed.

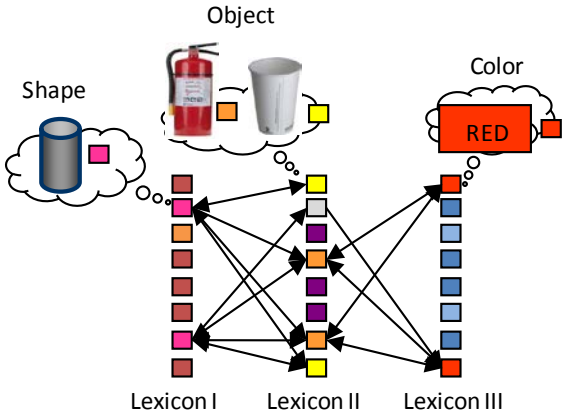


Figure 1: A simple example of lexicons, symbols and knowledge links

Figure 1 shows an example of lexicons, symbols and knowledge links. The three columns in Figure 1 represent three lexicons for the concept of shape, object and color with each box representing a neuron. Different combinations of neurons represent different symbols. For example, as Figure 1 shows, the pink neurons in lexicon we represent the cylinder shape, the orange and yellow neurons in lexicon II represent a fire extinguisher and a cup, while the red neurons in lexicon III represent the red color. When a cylinder shaped object is perceived, the neurons that represent the concepts "fire extinguisher" and "cup" will be excited. However, if a cylinder shape and a red color are both perceived, the neurons associated with "fire extinguisher" receives more excitation and become activated while the neurons associated with the concept "cup" will be suppressed. At the same time, the neurons associated with "fire extinguisher" will further excite the neurons associated with its corresponding shape and color and eventually make those symbols stand out from other symbols in lexicon I and III.

A computation model for cogent confabulation is proposed in [6]. Based on this model, a lexicon is a collection of symbols. A knowledge base from lexicon A to B is a matrix with the row representing a source symbol in A and a column representing a target symbol in B. The ijth entry of the matrix represents the strength of the link between the source symbol s_i and the target symbol t_j. It is quantified as the conditional probability $P(s_i \mid t_j)$. The knowledge bases are obtained during the learning procedure. During recall, the excitation level of all symbols in each lexicon is evaluated. Let l denote a lexicon, F_l denote the set of lexicons that have knowledge bases going into lexicon l, and S_l denote the set of symbols that belong to lexicon l. The excitation level of a symbol t in lexicon l can be calculated as: $I(t) = \sum_{k \in F_l} \sum_{s \in S_k} I(s) \left[\ln \left(\frac{P(s|t)}{p_0} \right) + B \right]$, $t \in S_l$. The function $I(s)$ is the excitation level of the source symbol s. Due to the "winner-takes-all" policy, the value of $I(s)$ is either "1" or "0". The parameter p_0 is the smallest meaningful value of $P(s_i \mid t_j)$. The parameter B is a positive global constant called the *bandgap*. The purpose of introducing B in the function is to ensure that a symbol receiving N active knowledge links will always have a higher excitation level than a symbol receiving (N-1) active knowledge links, regardless of the strength of the knowledge links.

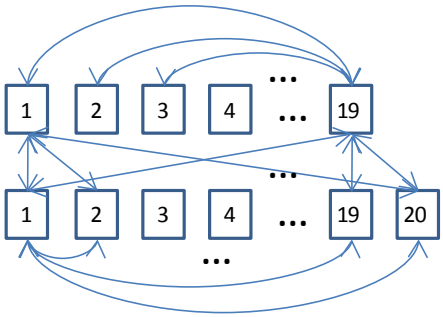


Figure 2: Lexicons and knowledge bases for confabulation based sentence completion (Not all knowledge bases are shown in the figure)

Based on the example provided in [6], confabulation based sentence and word completion software has been developed in AFRL/RIT. A sentence is represented using 39 lexicons that are arranged in 2 levels. The ith lexicon in level 1 represents the word (or punctuation) in the ith location of a sentence. Since there are 20 lexicons in level 1, the words or punctuations beyond the first 20 are discarded. The ith lexicon in level 2 represents two-word phrase that begins in the ith location of a sentence. The connections of the lexicons form a complete graph, i.e. there is a knowledge base between any two lexicons. Figure 2 shows the lexicons and the knowledge bases for the sentence completion problem.

4.1.2.3 **Hybrid model for Intelligent Text Recognition**

Military planning, battle field situation awareness, and strategic reasoning rely heavily on the knowledge of the local situation and the understanding of different cultures. A rich source of such knowledge is presented as natural-language text. In 2009, DARPA launched the Machine Reading program to develop a universal text to knowledge engine that scavenges digitized text to generate knowledge that can be managed by the artificial intelligence (AI) reasoning systems. The Machine Reading program limits its scope on the texts available on the World Wide Web. In real life, text exists in many forms other than its ASCII representation. These include printed texts such as books, newspapers and bulletins or hand written texts. There are many occasions when only the scanned or photographed image of the texts is available for computer processing. While the machine reading system bridges the gap between natural language and artificial intelligence, another bridge has to be constructed to link the natural existence of texts to their unique encoding that can be understood by computers.

Conventional Optical Character Recognition (OCR) tools or pattern recognition techniques are not enough to meet the challenges in this task. Because the text images are usually captured under extreme circumstances, sometimes the images will be noisy, or incomplete due to the damages of the printing material, or obscured by marks or stamps. Pattern recognition is extremely difficult, if not impossible, when the image is partially shaded or partially missing. For example, given the image in Figure 3, it would be impossible to recognize the characters that are smudged or missing using only image processing techniques. However, such task is not too difficult for human as we have anticipations for the missing information based on its context.

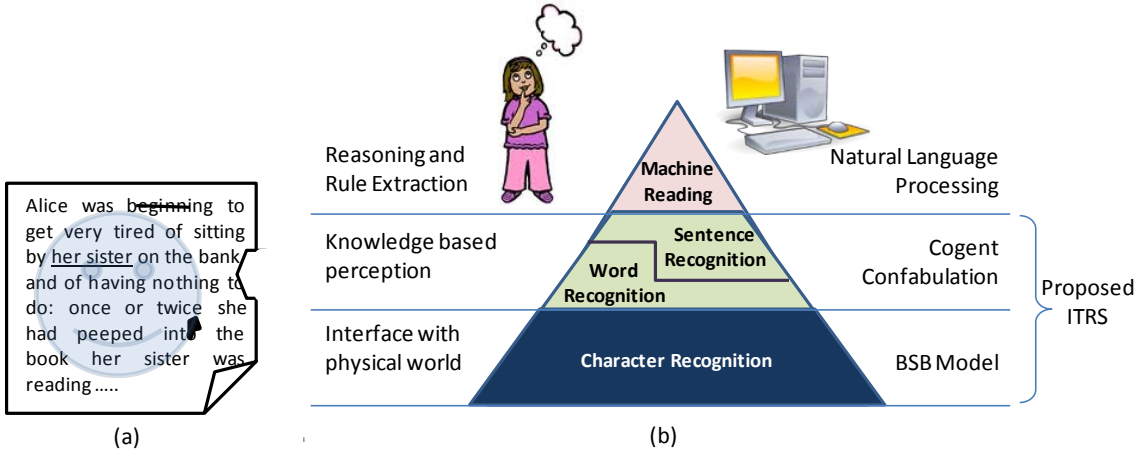


Figure 3: Lexicons and knowledge

In this project, we developed a prototype of context aware Intelligence Text Recognition System (ITRS) that mimics the human information processing procedure. The ITRS system learns from what has been read and, based on the obtained knowledge; it forms anticipations and predicts the next input image (or the missing part of the current image). Such anticipation helps the system to fight with all kinds of noises that may occur during recognition.

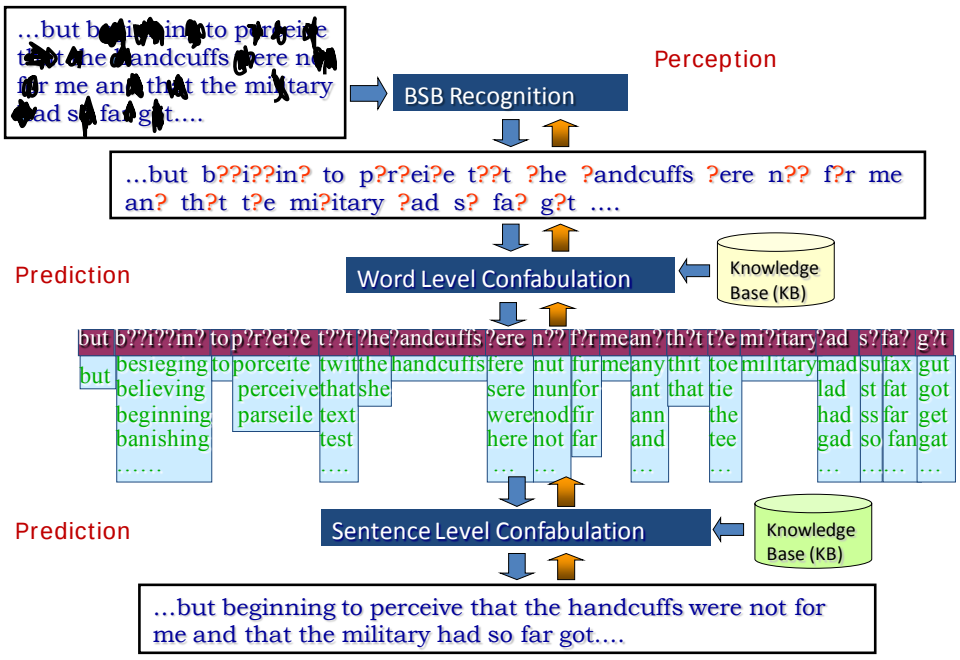


Figure 4: Hybrid model of ITRS

The ITRS is divided into 3 layers as shown in Figure 3. The input of the system is the text image. The first layer is character recognition software based on BSB models. It tries to recall the input image with stored image of English alphabet. In this work, a race model is adopted. The model assumes that the convergence speed of the BSB indicates the similarity between patterns. For a given input image, we consider all patterns that converge within 50 iterations as potential candidates that may match the input image. All potential candidates will be reported as the BSB results. Using the racing model, if there is noise in the image or the image is partially damaged; multiple matching patterns will be found. For example, a horizontal scratch will make the letter "T" looks like the letter "F". In this case we have ambiguous information.

The ambiguity can be removed by considering the word level and sentence level context, which is achieved in the second and third layer where word and sentence recognitions are performed using cogent confabulation models. The models fill in the missing characters in a word and missing words in a sentence. The three layers works cooperatively. The BSB layer performs the word recognition and it sends the potential letter candidates to the word level confabulation. The word recognition layer forms possible word candidates based on those letter candidates and send this information to the sentence recognition layer. There could be feedback paths that send the sentence level confabulation results back to word level or send word confabulation results back to character level. We believe that the feedback information can speed up the recognition process and its implementation will be our future task. Figure 4 shows an example of using the ITRS to read texts that has been smudged. The BSB recognize text images with its best effort. The word level confabulation provides all possible words that associates with the recognized characters while the sentence level confabulation finds the combination among those words that gives the most meaningful sentence.

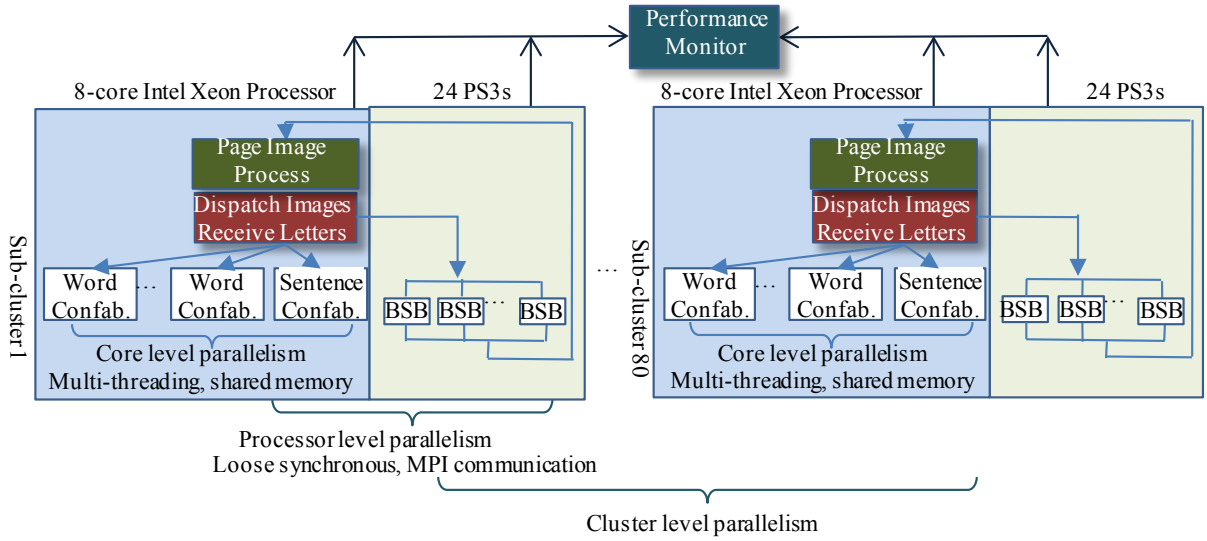


Figure 5: Software Architecture of ITRS system

4.1.3 Architecture of ITRS System on the Heterogeneous HPC Cluster

Because the 18,000 node HPC cluster was still not ready in the summer, we implemented the ITRS on a smaller cluster with similar structure. The cluster has 14 sub-clusters and each sub-cluster consists of one head node and 24 PS3s. The software architecture of the ITRS system is given in Figure 5. The architecture explores the parallelism in hardware and software to achieve a high throughput of the system.

We partition the entire workload into pages. All sub-clusters run simultaneously and independently to process different pages. In this way cluster level parallelism is achieved. There is a performance monitor that periodically checks the CPU utilization of the processors in the cluster for performance characterization. Because each sub-cluster loads pages on-demand, at cluster level, our system behaves asynchronously.

Upon receiving the page image, the head node will slice the image into small blocks, each one of these block contains one character. The blocks will be dispatched to the PS3s, where the BSB processes are running, for character reorganization. The results are sent back to the head node for word level and sentence level confabulation. With double buffering technique, the confabulation and BSB processes can be made parallel. Furthermore, all 132 SPEs in 22 PS3s are running simultaneously to process different characters. In this way, we achieve processor level parallelism. At this level, the system is loosely synchronous, because each BSB engine receives the same amount of image blocks and they perform the same amount of computation. Furthermore, because of the limited buffer space, periodic synchronization between the BSB and the confabulation is necessary. All inter-processor communication is implemented via Message Passing Interface (MPI).

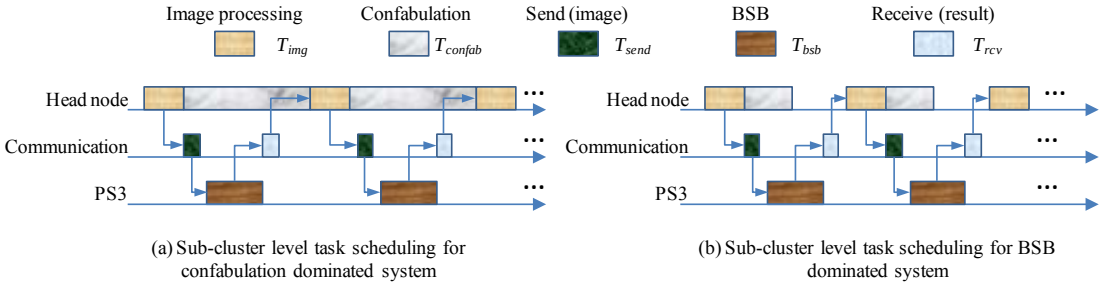
Based on the results from the BSB model, the host will fork multiple threads; each thread is a word level confabulation procedure. When all words in a sentence have been found, a sentence confabulation process is called. The word level and sentence level confabulation threads will be dispatched to different cores on the Intel Xeon processor, and in this way we achieve core level parallelism. The key reason that we choose thread level parallelism instead of process level parallelism is because it allows shared memory so that we do not have to duplicate the

knowledge base, which is more than 200MB large. In order to avoid frequent context switch, which usually happens when the number of threads is much greater than the number of cores, we adopt a token passing mechanism to control the number of threads. The program maintains a token pool. The number of tokens in the pool is less than or equal to the number of cores in the system. A token will be consumed when a thread is created and be returned when the thread ends. Because the threads are created on demand and complete dynamically, at this level, all cores work asynchronously.

4.1.3.1 **Sub-Cluster Level Task Interactions**

As we mentioned before, at sub-cluster level, the system is loosely synchronous. At the beginning of each iteration, the head node processes the scanned page and sends 96 character images to each PS3. Without waiting for the PS3 to send back the BSB results, the head node will go on working on the BSB results received in the previous iteration. During the same time, all PS3s performs the same amount of calculation and they will complete the calculation at approximately the same time. The potential candidates of matching patterns will be returned to the head node and stored in the MPI buffer. The head node will not process the MPI message until it has finished processing the previous returned results.

At sub-cluster level, two techniques are used to increase the throughput of the system. First, we pipeline the BSB model and confabulation model on PS3s and head node. Therefore, the throughput of the system is determined by the maximum delay of BSB and confabulation instead of the total delay of these two. Second, by using the MPI for inter-processor communication, we implicitly use double buffering technique to hide the communication latency.



(a) Sub-cluster level task scheduling for confabulation dominated system

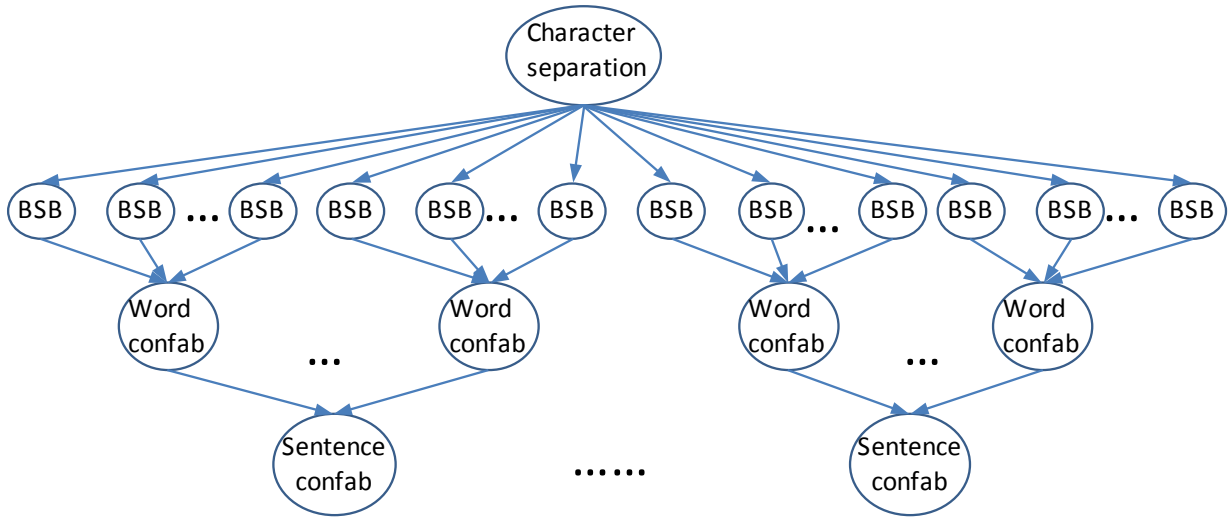
(b) Sub-cluster level task scheduling for BSB dominated system

Figure 6: Sub-cluster level task scheduling

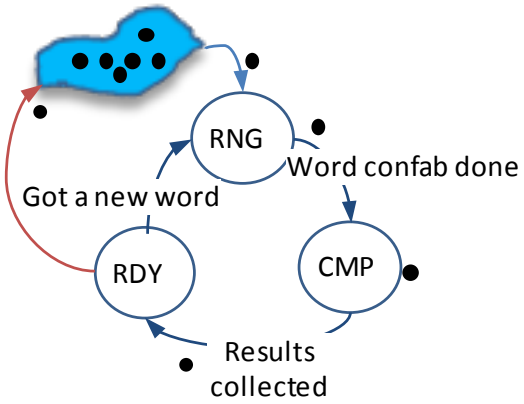
Figure 6 shows the sub-cluster level task scheduling. When the confabulation delay is much greater than the BSB delay, the communication latency for the send and receive procedure as well as the computation latency of the BSB model are hidden. The initiation interval of the system is determined by the delay of image processing and confabulation, i.e. $T = T_{img} + T_{confab}$. When the confabulation delay is less than the BSB delay, the computation latency of the confabulation model is hidden, the system initiation interval is determined by the delay of image processing, the communication delay of sending and receiving MPI messages and the delay of the BSB model, i.e. $T = T_{img} + T_{send} + T_{bsb} + T_{rcv}$. It is important to point out that T_{bsb} is a constant. For each input image, the same number of BSB models are evaluated. Each BSB model is run for the fixed number of iterations in order to check their convergence speed. The quality of input image does not affect the BSB computation time. However, a lower image quality means that more potential candidates will be found by the BSB. Therefore, it increases the workload and execution time for word and sentence confabulation.

4.1.3.2 **Multi-Threading Confabulation**

To fully utilize the multi-core architecture of the head node, layer 2 and 3 of the ITRS are implemented using multi-threading techniques. Figure 7 shows the task and data dependency graph of the ITRS system. A word confabulation process will not be triggered unless all the letters in that word have been processed by the BSB engine. Similarly, a sentence confabulation process will not be triggered unless all the words in that sentence have been confabulated. Obviously, the word confabulation process is triggered more frequently than the sentence confabulation process. Furthermore, each word confabulation takes longer time than sentence confabulation. This is because we need to find all valid words from the combinations of the letter candidates but only the most meaningful sentence from the combination of the word candidates.



(a) Task graph of the ITRS system



(b) State machine for Token management

Figure 7: Task graph of the ITRS and the token management for multi-threading system

The above analysis shows that in order to maximize the throughput, it is necessary to parallelize the word confabulation processes. Figure 8 shows how the ITRS tasks are mapped to a sub-cluster with 24 cell processors and one N-core head node. At anytime, on the N-core head node

we could run 1 thread of sentence confabulation and N-1 threads of word confabulation. Each word confabulation thread processes one word. The sentence confabulation thread is the main thread which is always active. Besides sentence confabulation, it also performs other tasks such as character seperation and communicates with the BSB engines. The word confabulation thread is dynamically created when all letters belonged to a word has been processed by the BSB engine. After the word confabulation completes, the thread will be deleted.

We keep the number of thread to be exactly equal to the number of cores in the processor in order to avoid excessive context switch. This is achieved by using a token passing mechanism. A token is used to represents the status of a core. It can be in 3 states: ready state, running state and completion state. A token pool is maintained in the main thread. The number of tokens in the pool equals to the number of cores in the system. All token in the pool is in the ready state. When all letters of a word are received and if there is a token in the pool, a new word confabulation thread is created and a token is removed from the pool. The status of that token is changed to "running". When the word confabulation completes, the thread is deleted and the token state is changed to "completion." Only after the results of the word confabulation have been collected by the main thread, the status of the corresponding token will be changed to "ready" and the token will return to the pool. The token passing mechanism guarantees that at any time the number of active thread is no more than the number of cores.

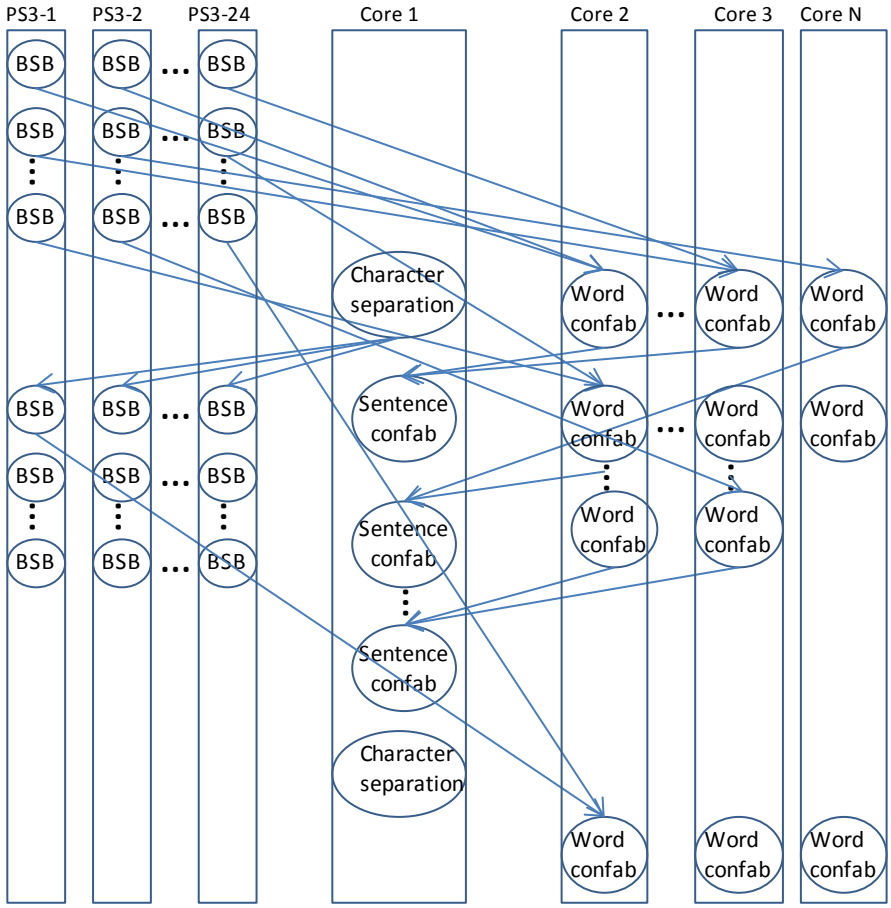


Figure 8: Mapping ITRS tasks to a cluster with 24 PS3 processors and an N-core head node

The multi-threading architecture leads to an interesting synchronization problem. As shown in Figure 9, two circular buffers are maintained in the main thread. The first buffer is referred as "input buffer". It stores the output from the BSB engine which will be used as the inputs by word confabulation. The input buffer is written in sequential order. It is read by the word confabulation also in sequential order. The results from word confabulation will be written into an "output buffer". The item in the *i*th location of the input buffer will be written into the *i*th location of the output buffer. There are up to N-1 threads of word confabulation working simultaneously on N-1 different words. Their processing speeds are different. Whenever a thread completes processing a word, it writes the result to the corresponding location in the output buffer and fetches another word from the input buffer. As a result, the output buffer will be written out-of-order. The output buffer will be read by sentence confabulation process again in sequential order. A read pointer is used to indicate the starting word of the next sentence. When the next M entries from the read pointer have been filled (M is the number of the words in the next sentence), the sentence confabulation process will be called and those entries will be removed from the output buffer.

In general, a new thread of word confabulation will start as long as the input buffer is not empty and a token is available. However, due to the variable confabulation speed of different words, it is possible that one of the threads is still working on a word that belongs to the sentence that will be read out in the next, while other threads have already filled up the rest of the buffer. Because the output buffer must be read out in sequential order, no sentence can be read from the buffer before the current sentence is read. In this scenario, the output buffer is "full" and a stall happens. No new word will be fetched from the input buffer until the next sentence is removed from the output buffer. More strictly speaking, the stall happens when the following 3 conditions are true:

1. The read pointer of the output buffer is at the ith location,
2. There is one word confabulation thread working on the jth location, $j - i < M$, where M is the number of words in the next sentence,
3. The current read pointer of the input buffer is at location j-1.

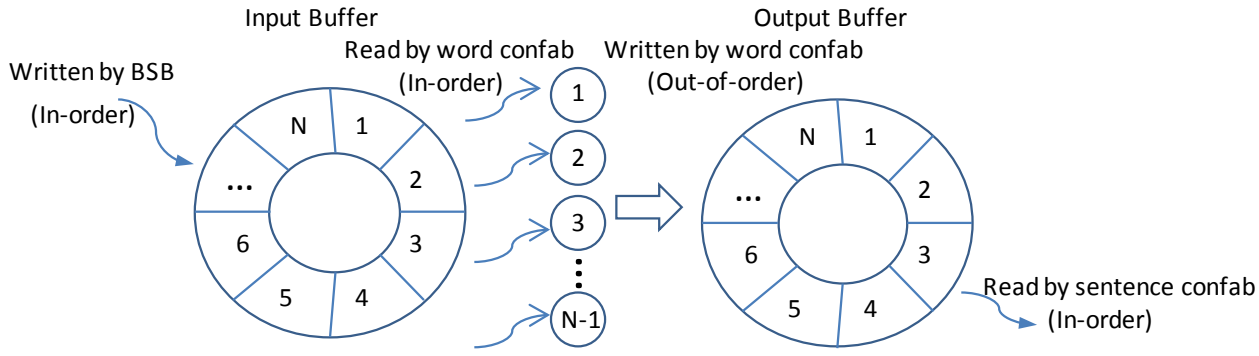


Figure 9: An in-order/out-of-order double buffering system

The stall is used to synchronize the speed of different word confabulation processes; therefore we refer to the delay that is introduced by the stall as synchronization delay. The synchronization delay increases when the variation of the word confabulation time gets larger.

4.1.3.3 **Character Separation**

The flow of page image processing is given in Figure 10. The page image is initially in JPG format. It is converted into BMP format. The page is first divided horizontally into different lines and each line is vertically divided into characters. The bit map of each character will be stretched or scaled into 15X15 pixels. Finally the grey scale value of each pixel will be discretized into binary level, which is the required format for the BSB model.

4.1.3.4 **Performance Monitor**

A performance monitor is implemented to monitor and display the CPU activities of each sub-cluster. Figure 11 shows the graphic display of the performance monitor. The top bar represents the Xeon head node. Each small box represents a single core in the head node. The 24 square boxes represent 24 cell processors. The upper rectangle in the box represents the PPE in the cell processor. The 6 smaller boxes below it represent the 6 SPEs in the same processor. The activities of processors are color coded with red representing the highest activity and blue representing the lowest activity.

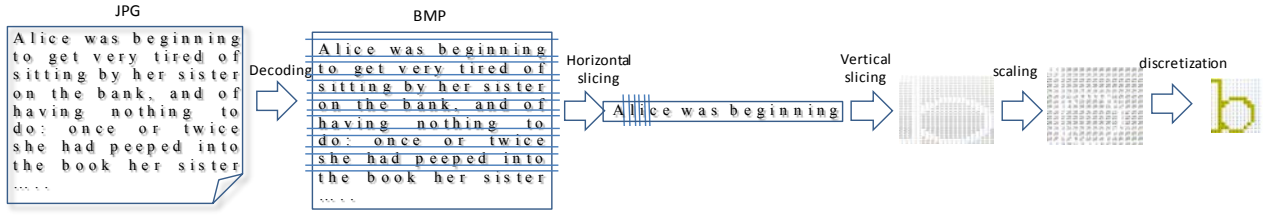


Figure 10: Processing flow of page image processing

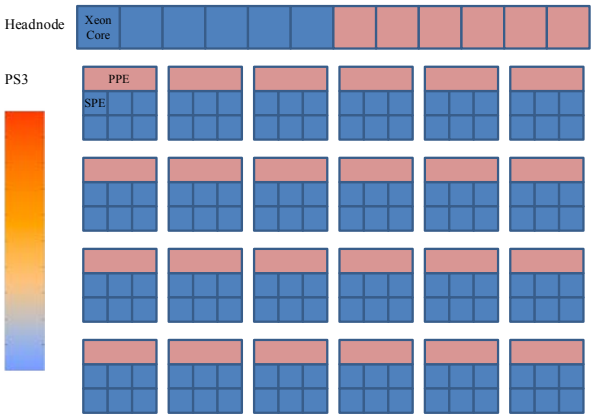


Figure 11: Graphic display of performance monitor

4.1.4 Experiment Results

We evaluate the ITRS system for its performance and accuracy. Its word level knowledge is trained by reading an English dictionary and its sentence level knowledge base is trained by reading 70 classic literatures. Our test case is extracted from the book "Great Expectation" by Charles Dickens. The text consists of 96767 letters or 23912 words. The text has not been read during the training process. In order to explicitly control the noise in the input, we use generated

bit map of text images instead of scanned text images. Horizontal scratches are added to the images of letters selected randomly. The amount of noises in the input is controlled by two means. (1) The thickness of horizontal scratches varies from 1 pixel to 3 pixels. Figure 12 shows the examples of the three different types of horizontal scratches. (2) The probability of scratched characters varies from 0.2, 0.4 to 0.6.

Figure 12: Three different horizontal scratches

Table 1 (a) and (b) give the rate of accurate sentence and accurate words after confabulation. They are calculated as the number of sentences (words) that have been correctly confabulated divided by the number of sentences/words that have scratches on it. Table 1 (c) gives the overall rate of correct words. It is calculated as the total number of correct words (including both confabulated and none-confabulated) divided by the total number of words in the text. The data in Table 1 (c) and (b) are very close to each other. This is because the majority of words have at least one scratch. Therefore, they all need to go through the word confabulation process.

Table 1: Recall accuracy at sentence and word level

(a) Sentence confabulation

Scratch / Probability	1	2	3
0.2	0.983051866	0.97748934	0.967612
0.4	0.977124818	0.965347529	0.950872
0.6	0.970695971	0.953397447	0.928233

(b) Word confabulation

Scratch / Probability	1	2	3
0.2	0.92006	0.902866	0.862852
0.4	0.86588	0.820211	0.764404
0.6	0.818703	0.742383	0.649774

(c) Total accurate word

Scratch / Probability	1	2	3
0.2	0.989694866	0.986881052	0.981102
0.4	0.982713059	0.97478531	0.96459
0.6	0.975735428	0.962104878	0.942372

Figure 1 shows the rate of correct sentence and correct words found by the ITRS when a well trained sentence level knowledge base (i.e. "long KB") is used and a poorly trained sentence level knowledge base (i.e. "short KB") is used. The size of the high quality knowledge base ("long KB") is more than 6 GB, while the size of the low quality knowledge base ("short KB") is 2.7 MB. The data series labeled as "% improve" gives the percentage improvement of the results obtained using "long KB" over the results obtained using "short KB". The results show that better knowledge at sentence level improves the sentence accuracy up to 80% and word accuracy up to 8%.

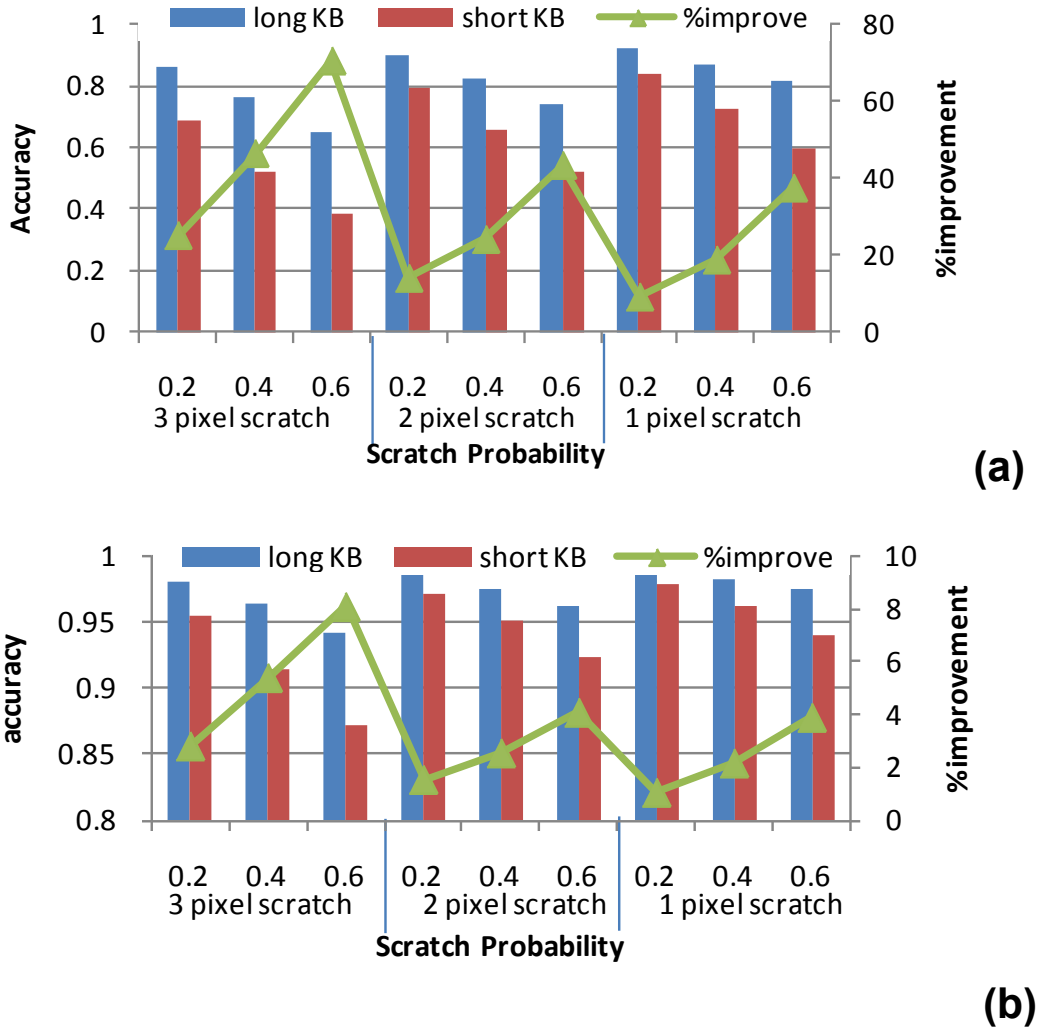


Figure 13: Impact of sentence level knowledge base on the confabulation accuracy: (a) Comparison in sentence accuracy and (b) Comparison in word accuracy

Figure 14 shows the relation between the word and sentence accuracy versus the level ambiguity in the input. The ambiguity is calculated using four different ways. "Cand./L" gives the average number of letter candidates for each letter in the input text, "candid/XL" gives the average number of letter candidates for each scratched letter in the input text, "candid/W" gives the

average number of candidate words for each word in the input text, and "candid/XW" gives the average number of candidate words for each word that has letters being scratched. The results show that the text recognition accuracy is a linear function with "candid/L" and "candid/W". However, it has low correlation with "candid/XL" and "candid/XW". The results indicate that the accuracy of ITRS is not determined by how many letters/words are scratched out and how severely each letter/word is scratched. It is determined by the ratio of the amount of ambiguous information versus the total available information. In other words, the ITRS performs letter/word recognition not only by looking at the ambiguous information itself by also by considering the context of the ambiguous information.

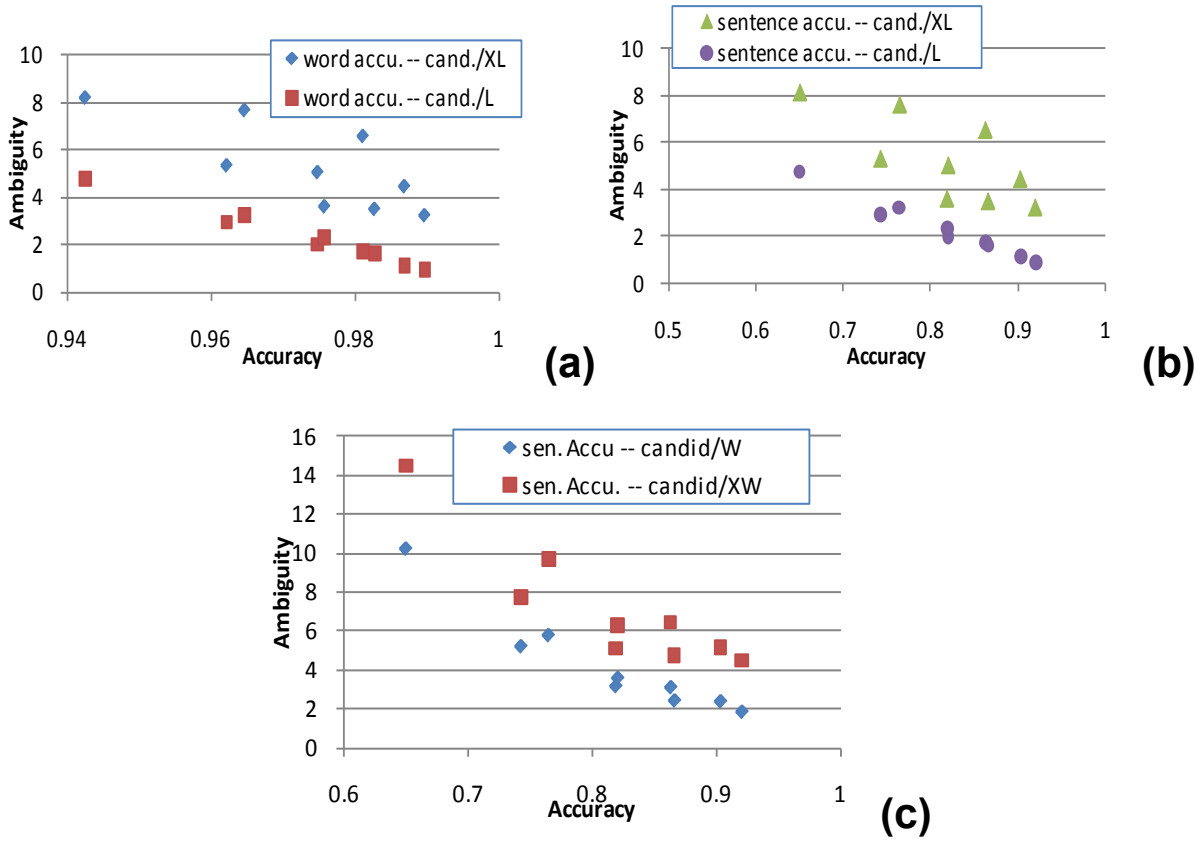


Figure 14: Word/sentence level accuracy versus the ambiguity: (a) Word accuracy vs. letter ambiguity, (b) (b) Sentence accuracy vs. letter ambiguity, and (c) (b) Sentence accuracy vs. word ambiguity

Figure 15 shows the history of how the ITRS software architecture evolves during this summer. Before the summer, we have a base line implementation of the system. The architecture is shown in Figure 16 (a). All the software components are connected sequentially in an ad-hoc way except for the BSB engines that are running on 24 PS3s in parallel. Our first step is to improve the confabulation speed by multi-threading. Figure 16 (b) shows the improved architecture.

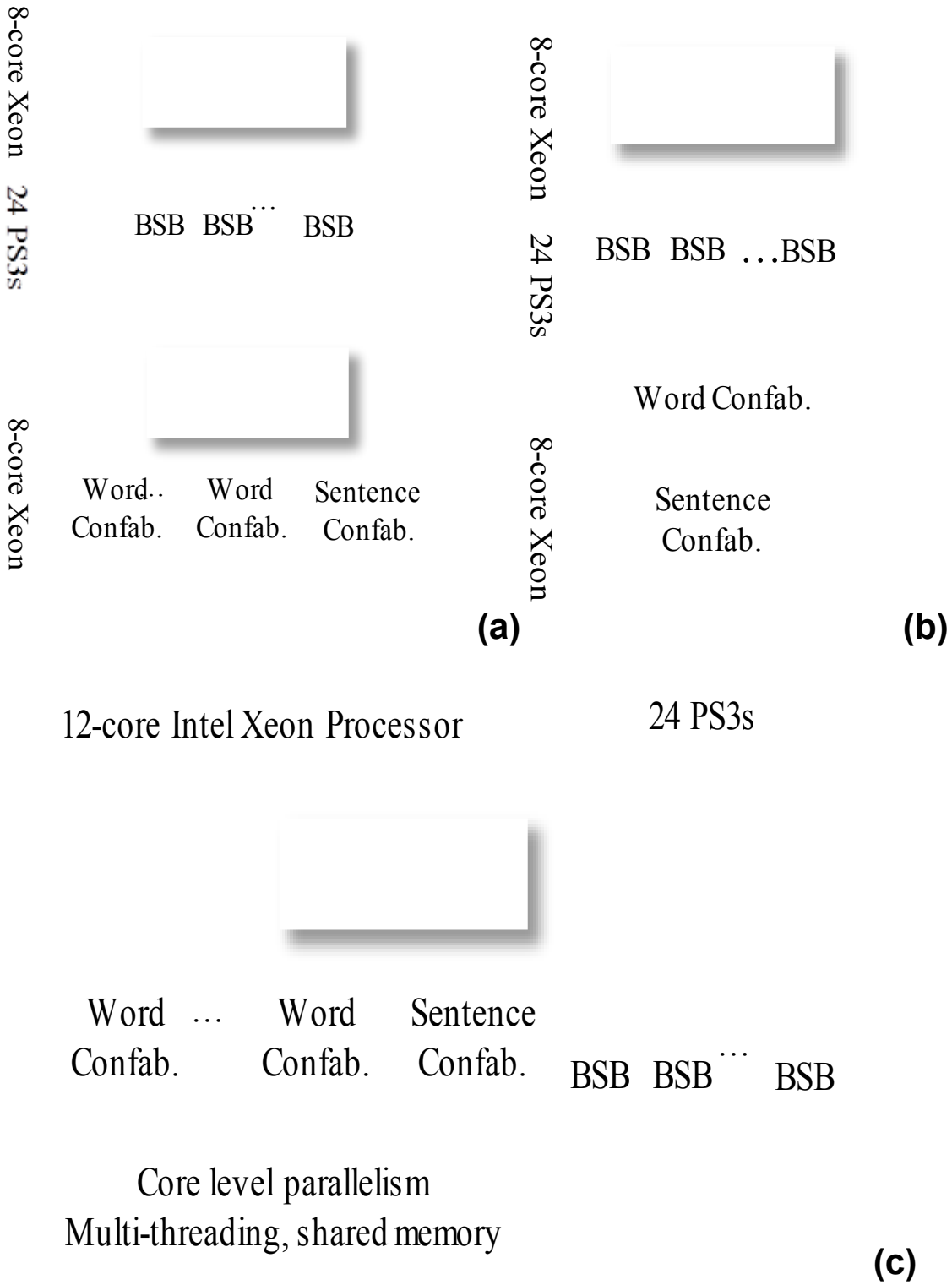


Figure 15: Improvement of the ITRS system architecture

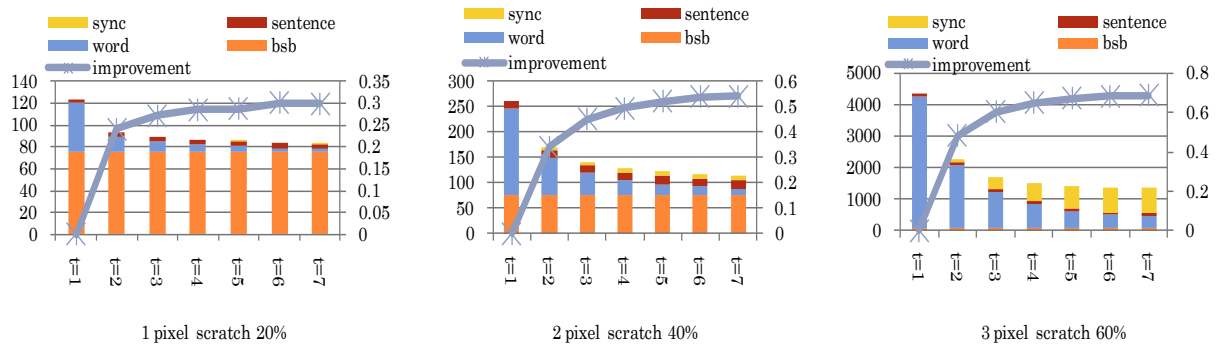


Figure 16: Performance improvement by multi-threading confabulation

To evaluate the performance of the improved architecture, we carried out experiments on 3 different input test cases. In the first input file 20% of character images are scratched by 1 pixel wide horizontal bars. Compared to the other two test cases, it has the highest image quality. The second input file has 40% of character images scratched by 2 pixel wide horizontal bars. Compared to test cases 1 and 3, it has the medium image quality. The last input file has 60% of character images scratched by 3 pixel wide horizontal bars. It is referred as the lowest quality input file. The number of word confabulation threads is varied from 1 to 7 and denoted as *t*. We recorded the total runtime. The runtime is further broken down into BSB time, word confabulation time, sentence confabulation time and synchronization time. The concept of synchronization delay is introduced in section 3.2. The size of the input/output buffer in the double buffering system is set to 100 sentences.

Figure 16 shows the runtime information for the 3 test cases when the number of word confabulation threads increases from 1 to 7. It also reports the performance improvements of the multi-threading implementations compared to the baseline implementation.

Several observations can be obtained from the results:

1. No matter how the image quality changes, the BSB time remains constant.

2. When the quality of the input text image deteriorates, the word/sentence confabulation time increases. This is because we rely on the confabulation to resolve the ambiguity in the input.

3. When the quality of the input text image deteriorates, the synchronization delay gets longer. This is because the variations in the word confabulation speed increases as the level of ambiguity rises, and the in-order/out-of-order circular buffer will be blocked more frequently.

4. With multi-threading technique, we can improve the runtime by up to 70% compared to the base line implementation.

The results in Figure 16 show that with low quality input, the synchronization delay becomes the bottleneck that prevents us achieving linear speedups by using multi-threading technique. One way to relieve this bottleneck is to increase the capacity of the double buffering system. We increase the buffer size from 100 sentences to 200 and 300 sentences and run the experiment again on the low quality input file. Figure 17 gives the runtime information for the systems with 3 different buffer configurations. The last data series (i.e. “buffer imprv”) gives the performance improvement due to the increased buffer size. The results show that with 7 word confabulation

threads, increasing the buffer size from 100 to 200 and 300, we reduce the runtime by 20% and 30% respectively.

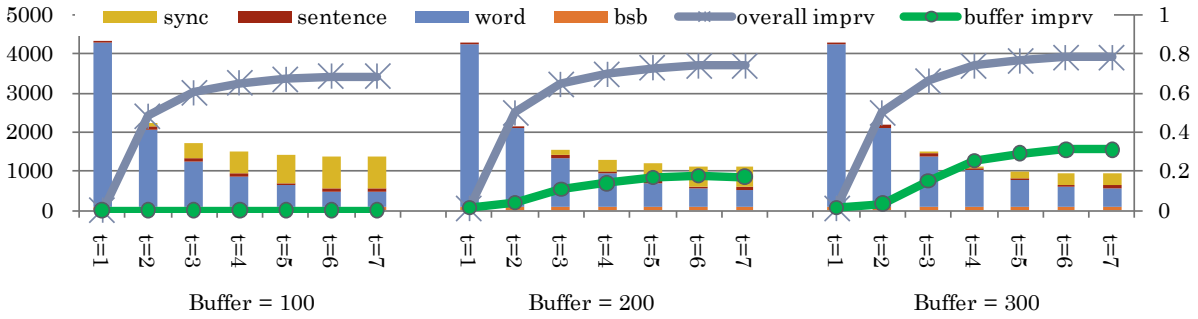


Figure 17: Increasing the buffer size reduces the synchronization delay

We further improve the software architecture of the ITRS by parallelizing the BSB and confabulation processes. The new architecture is shown in Figure 15 (c). Figure 18 shows the performance of the improved system on high quality, medium quality and low quality inputs. The buffer capacity is set to 300 sentences. The data series labeled "improvement" gives the performance improvement of the system over the base line implementation, while the data series labeled "improv2" gives the percentage speed improvement by comparing the parallel ITRS with multi-threading ITRS. The number of word confabulation threads and the buffer size of these two systems are kept the same. The results show that parallelizing the BSB and confabulation is most effective for the medium quality test cases, because the BSB time and confabulation time are approximately equal for this type of test cases and execute them simultaneously can cut the total runtime by 50%.

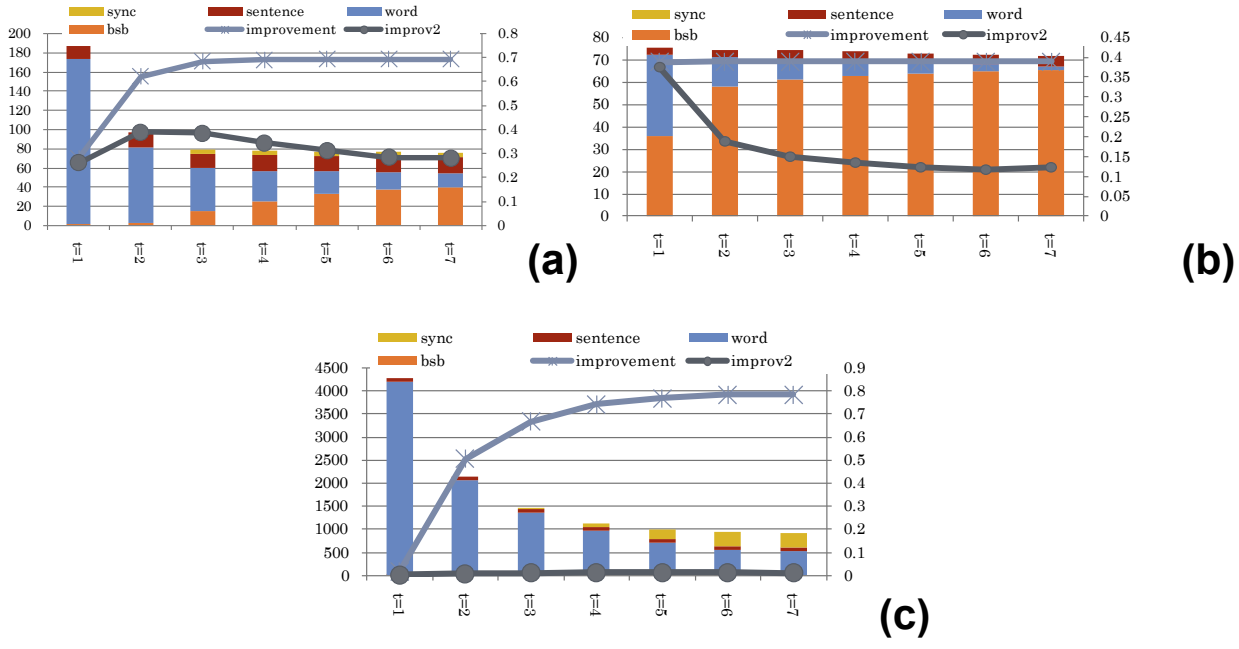


Figure 18: Performance improvement by parallelizing BSB and confabulation: (a) Results for high quality test case, (b) Results for medium quality test case, and (c) Results for low quality test case

The previous experiments clearly show that the runtime of ITRS increases when the quality of the input text image reduces. The quality of the input text image can be measured by a parameter called "level of ambiguity." It is calculated as the average number of word candidates for each input word. Figure 19 shows the relation between the time for ITRS to read one page of text image, which consists of 4000 letters, and the level of ambiguity in the input image. As we expected, when the information ambiguity increases, the ITRS read time increases exponentially. In this experiment, the number of word confabulation threads is set to 1.

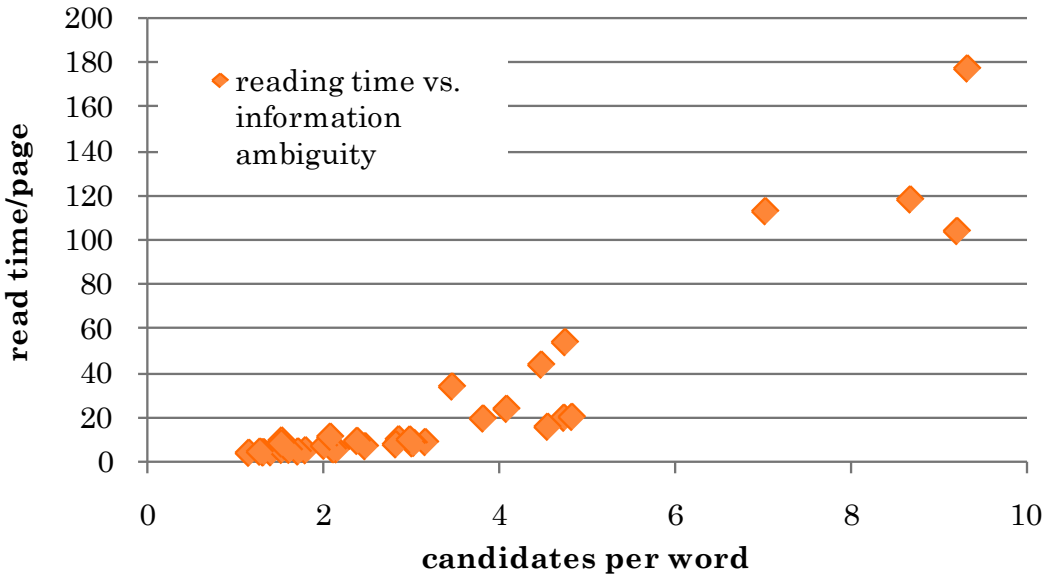


Figure 19: The ITRS read time increases exponentially as the level of ambiguity in the input increases

4.1.5 Conclusions

In this work, we introduced the hardware and software architecture of the Intelligent Text Recognition System. The system is based on a neuromorphic computing model. To achieve high performance we explore the parallelism in the hardware and the software. Detailed performance analysis is provided in this report. The results show that the ITRS system can achieve more than 90% accuracy at sentence level and more than 98% accuracy at word level.

5.0 REFERENCES

[1] R. Wray, C. Lebiere, P. Weinstein, K. Jha, J. Springer, T. Belding, B. Best, and V. Parunak, "Towards a Complete, Multi-level Cognitive Architecture," in Proc. of the International Conference for Cognitive Modeling, 2007.

[2] J. A. Anderson, J. W. Silverstein, S. A. Ritz, and R. S. Jones, "Distinctive features, categorical perception, probability learning: Some applications of a neural model," in Neurocomputing; Foundations of Research, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA: The MIT Press, 1989, ch. 22, pp. 283–325, reprint from Psychological Review 1977, vol. 84, pp. 413–451.

[3] "Associative Neural Memories: Theory and Implementation," Mohamad H. Hassoun, Editor, Oxford University Press, 1993.

[4] A. Schultz, "Collective recall via the Brain-State-in-a-Box network," IEEE Transactions on Neural Networks, vol. 4, no. 4, pp. 580–587, July 1993.

[5] Q. Wu, P. Mukre, R. Linderman, T. Renz, D. Burns, M. Moore and Qinru Qiu, "Performance Optimization for Pattern Recognition using Associative Neural Memory," Proc. Of 2008 IEEE International Conference on Multimedia & Expo, June 2008.

[6] R. Hecht-Nielsen, "Confabulation Theory: The Mechanism of Thought", Springer, Aug. 2007

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

AFOSR	Air Force Office of Scientific Research
AFRL	Air Force Research Laboratory
BOLD	Blood oxygen level dependent
BSB	Brain State in a Box
CELL-BE	Cell Broadband Engine
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency
fMRI	functional magnetic resonance imaging
FTR	Final Technical Report
HPC	High Performance Computer
ITRS	Intelligent Text Recognition System
JBI	Joint Battlespace Infosphere
JPG	Joint Photographic Group
MPI	Message Passing Interface
OCR	Conventional Optical Character Recognition
Ops	operations
OS	Operating System
PDES	Parallel Discrete Event Simulation
PPE	Parallel Processor Engine
PS3	Sony PlayStation3
RI	Information Directorate
SPE	Synergistic processing elements
TFLOPs	Tera FLoating Point Operations Per Second